Software Development Patterns For Startups

Jorge A. Melegati, Alfredo Goldman
{melegati,gold}@ime.usp.br
Instituto de Matemática e Estatística
Universidade de São Paulo

Abstract: Technology startups are recently founded companies focused on innovation, generally, under lack of resources. Then, a highly uncertain environment is created that influence how they develop their software. This set of patterns tries to capture the expertise found in literature to help founders and employee developers to optimize software development under these unique circumstances that a startup faces. Until incipient, we could present five new patterns (Be Agile, Use Open Source Solutions, Simple Tools, Validate your Hypothesis with Early Adopters and Empower Team Members or Have a Flexible Structure) based on literature.

1. Introduction

Technology startups are present in innovation environment and in economy, in a general sense, since the 1970s. Nevertheless, like agile methodologies, their suggested best practices are being discussed after being applied in industry, inverting the natural path from scientific research to industry. As a matter of fact, efforts to reach best practices on developing startups came to light after the "dot com bubble", at the beginning of years 2000, which led several young companies to close and investments in the sector to fall abruptly.

This set of patterns tries to formalize several practices described in literature to improve software development, specifically inside tech startups, aiming better processes and products of these companies. To classify our patterns, we used a classification proposed by Paternoster et al. in their systematic mapping about software development in startups [3]: process management practices, software development practices (subdivided in requirement engineering practices, implementation, maintenance and deployment practices, quality assurance practices), managerial and organizational practices and tools and technologies. In section 2, we review works on literature related to our subject and then a work-in-progress set of patterns is presented, in section 3 more detailed and in section 4 just names of proposed patterns.

2. Related work

Hecksel [4], in work published at PLoP'04, propose a set of patterns to software development in general focusing on the following topics: methodology, methodology selection, team composition, management and project management. Several patterns present in this work could be used for problems in startups, some with minor modifications and others in the same way they are presented. In this set, some represent good advices to startups like Utilize Tools Over Growth, Set Senior Developer Ratio and Breadth Over Depth. Under methodology category, Utilize Tools Over Growth gives preference to use productivity tools "when there are low hanging fruits manual tasks that can provide large return on investments if automated". This is especially true in a startup since financial resources to hire are scarce. But there is a trap here that should be avoided using Set Senior Developer Ratio. When hiring, startup founders tend to try to save money then search for junior developers, however Hecksel [4] advocate for a minimum rate of 25% of senior developers up to 50%, since, he argues citing Pragmatic Programmer from Dave Thomas that "adding a Senior Developer to the project team can have up to a 20x productivity contribution to the team compared to adding another inexperienced team member" but too many senior developers is also bad because it will create the "all chiefs no Indians" effect where expectations about decision making authority would not be satisfied for all. This pattern is under team composition category from we can discuss the Breadth Over Depth pattern that advocates hiring team members that understand the "big picture on both project requirements and selected technologies" instead of outstanding developers that only master a single technology. In a startup, this makes total sense since everything, from processes to the software itself, have to be built and help from all is very useful and, because of the high uncertain environment, some members could leave the company and it will slow the development if any of them is the only one who dominate a technology important for the development.

Although not focusing directly on startups, some pattern languages found in literature could indicate good practices that could be used in these young companies, either with some adaptations or even the way that were proposed for mature teams or companies. Azua et al. [5] create a pattern language for release and deployment process because, according to the authors, changes in software systems must occur in a controlled and disciplined way. The language would help software delivery be controlled and disciplined in an agile environment. Hilst e Fernandez [6] analyze theories behind most used practices in process improvement, among them CMMI, Lean and Agile, extracting a set of patterns to approach the theories that support these methodologies even these theories not being well understood. The set is composed of five patterns (Have a Plan, Copy What Works or Better Practices, Remove Waste or Flow, Consider All Factors or System Thinking and Incorporating Feedback and Learning) and four antipatterns (And a Miracle Occurs, Buy a Silver Bullet, All Problems are Nails and Solutions must be General). Ernst [7] use patterns approach to discuss the enterprise architecture problem (EA) that aim align information technology area with business of a company, which is very important for startups since both areas development is done simultaneously in these companies that, many times, are still building a new market. Rikner and West [8] argue that developers are designers since "they design everything from user interface to program architectures to algorithms" and, besides that, receive little education on design discipline so, a set of patterns that represent designers way of thinking, called design thinking, is presented.

3. Patterns

Pattern: Be Agile

Category: Process Management

Context: In the beginning of years 2000, it occurred the so-called "dot-com bubble" phenomenon: the price of stocks of several Internet companies fell abruptly leading to big losses for investors and total loss of interest in these companies for investing more money, but these companies still needed money because of their lack of revenue and dependence on new investments. These companies used the traditional product development method in which a new product is released to public after a time and money consuming development phase. This product development approach was severe criticized by Blank [1] who created a new methodology to gather users feedback as soon as possible. Ries [2], Blank's student, extended this vision to other areas of a startups besides product development like marketing and created Lean Startup, a set of practices to create products/companies based on innovation in high uncertainty environments. According to Ries, the development of a new product should be based in hypothesis that should be validated as soon as possible. For that, he uses a MVP (Minimum Viable Product) which is a draft product created with just the necessary to test a hypothesis and receive user feedback. In this environment, since requisites change frequently, it is obvious that traditional software development techniques are not adequate.

Problem: A startup, by being in an innovator and uncertain environment, will have several changes during its product development, that is, software requisites will be volatile. For a startup to have an organized software development, it will need a methodology that supports change instead of avoid it.

Forces: The forces present here are predictability and responsiveness to change. Startups should be trustworthy to its stakeholders, like clients, because they could not be patient to failures as the product is new, and investors who support the company while it could not stand by itself. Nevertheless, they could not be plastered to changes because this is not compatible to the environment they are in and it is not what is expected from them.

Solution: Agile methodologies, like XP and Scrum, are considered processes best suited to software startups since they embrace change instead of avoid it, like cited by Paternoster [3].

Applicability: Software development startups, generally, are subject to product changes and these should be implemented quickly, so agile methodologies should be used, except when specific conditions related to certification processes are required like in pharmaceutical or defense markets.

Related Patterns: We did not find any related pattern.

Pattern: Use Open Source Solutions

Category: Tools and Technologies

Context: The startup definition is not a consensus in literature. Different authors use different definitions, as observed by Paternoster [3], although some common elements are present in definitions like limited human and financial resources. Startups also, usually, build brand-new products disrupting some market and may have special needs for their libraries or other third party software on which its product is built, because of that, developers should be able to adapt or personalize this third party software.

Problem: Startups do not have money cannot afford software solutions, that support their software development, with good quality and support and do not have human resources to develop them in-house either.

Forces: Startups should make products of good quality to win the market which is composed of clients always more demanding and flaws sensible while struggle to survive under lack of resources.

Solution: Open source solutions are a natural way for startups since they do not have cost acquire them and, if well chosen, through active communities, could have support of quality and, again, without cost.

Applicability: Generally, startups use several open source solutions but some exceptions may exist. Areas extremely sensible to flaws like health related could not accept the use of some piece of code that does not have warranty over it like most of open source code is.

Related Patterns: Simple tools.

Pattern: Simple Tools

Category: Tools and Technologies

Context: Like said before, startups live with lack of resources, humans and financial, so, tools used for process management (e.g., process management software) and, even, software development itself (e.g., CASE tools), cannot be complex as in consolidated companies because they demand more investment and time to implement.

Problem: Lack of human and financial resources and time- prevent a startup to use complex tools that could leverage their development productivity.

Forces: As previous discussed, startups have to respond quickly to changes in their market and, for that, must use their resources the most optimized possible way because of the lack of them, although, this same lack of resources prevent complex productivity tools.

Solution: In startups, simple tools are preferred both to process management and internal communication, using white boards, cards and general use instant messaging software, and for software development like simple software environments instead of more complex tools.

Related patterns: Use open source solutions – several tools used by startups are open source.

4. Possible Patterns

Pattern: Validate your Hypothesis with Early Adopters

Category: Software development practices – Implementation, maintenance and deploy

Pattern: Empower Team Members or Have a Flexible Structure

Category: Managerial and Organizational practices

5. Conclusion

A list of possible patterns for software development in startups was presented and discussed and others were just presented. It is expected that a deeper research should bring others patterns to light even because not all categories proposed by Paternoster were used for the already presented patterns. However, it was possible to discuss some important points on the subject like startup definition but others could appear after more discussion.

References

[1] Blank, S. Four Steps to the Epiphany, 2005

[2] Ries, E. Lean Startup, 2011

[3] Paternoster et al. Software development in startup companies: A systematic mapping study, Information and Software Technology, 2014

[4] Hecksel, D. Software Development Patterns, in: 11th Conference on Patterns Languages Of Programs, PLoP, 2004

[5] Azua, M., Bygrave, D., Rodin, R., Sadovski, E.. A pattern language for release and deployment management, in: Proceedings of the 18th Conference on Pattern Languages of Programs, 2011

[6] Van Hilst, M., Fernandez, E. B. A pattern system of underlying theories for process improvement, in: Proceedings of the 17th Conference on Pattern Languages of Programs, 2010

[7] Ernst, A. M. Enterprise Architecture Management Patterns, in: Proceedings of the 15th Conference on Pattern Languages of Programs, 2008

[8] Rikner, R., West, D. Design Thinking Patterns, in: Proceedings of the 17th Conference on Pattern Languages of Programs, 2010

[9] Coleman e O'Connor An investigation into software development process formation in software start-ups

[10] Sutton et al. The Role of Process in a Software Start-up